What Is Claimed Is:

1　　1.　　A method for executing a commit instruction to facilitate
2　transactional execution on a processor, comprising:
3　　　　encountering the commit instruction during execution of a program,
4　wherein the commit instruction marks the end of a block of instructions to be
5　executed transactionally; and
6　　　　upon encountering the commit instruction, successfully completing
7　transactional execution of the block of instructions preceding the commit
8　instruction;
9　　　　wherein changes made during the transactional execution are not
10　committed to the architectural state of the processor until the transactional
11　execution successfully completes.

1　　2.　　The method of claim 1, wherein successfully completing the
2　transactional execution involves:
3　　　　atomically committing changes made during the transactional execution;
4　and
5　　　　resuming normal non-transactional execution.

1　　3.　　The method of claim 2, wherein atomically committing changes
2　made during the transactional execution involves:
3　　　　treating store-marked cache lines as locked, thereby causing other
4　processes to wait to access the store-marked cache lines;
5　　　　clearing load marks from cache lines;

6   committing store buffer entries generated during transactional execution to

7   memory, wherein committing each store buffer entry involves unmarking, and

8   thereby unlocking, a corresponding store-marked cache line; and

9   committing register file changes made during transactional execution.

1   4.   The method of claim 1, wherein if an interfering data access from

2   another process is encountered during the transactional execution and prior to

3   encountering the commit instruction, the method further comprises:

4   discarding changes made during the transactional execution; and

5   attempting to re-execute the block of instructions.

1   5.   The method of claim 1, wherein for a variation of the commit

2   instruction, successfully completing the transactional execution involves:

3   atomically committing changes made during the transactional execution;

4   and

5   commencing transactional execution of the block of instructions following

6   the commit instruction.

1   6.   The method of claim 1, wherein potentially interfering data

2   accesses from other processes are allowed to proceed during the transactional

3   execution of the block of instructions.

1   7.   The method of claim 1, wherein the block of instructions to be

2   executed transactionally comprises a critical section.

1   8.   The method of claim 1, wherein the commit instruction is a native

2   machine code instruction of the processor.

23

Attorney Docket No. SUN-P9324-MEG   Inventors: Tremblay et al.

ARP H:\SUN MICROSYSTEMS\SUN-P9324-MEG\SUN-P9324-MEG APPLICATION.DOC

1   9.   The method of claim 1, wherein the commit instruction is defined

2   in a platform-independent programming language.

1   10.   A computer system that supports a commit instruction to facilitate

2   transactional execution, wherein the commit instruction marks the end of a block

3   of instructions to be executed transactionally, comprising:

4   a processor; and

5   an execution mechanism within the processor;

6   wherein upon encountering the commit instruction, the execution

7   mechanism is configured to successfully complete transactional execution of the

8   block of instructions preceding the commit instruction;

9   wherein changes made during the transactional execution are not

10   committed to the architectural state of the processor until the transactional

11   execution successfully completes.

1   11.   The computer system of claim 10, wherein while successfully

2   completing transactional execution, the execution mechanism is configured to:

3   atomically commit changes made during the transactional execution; and

4   to

5   resume normal non-transactional execution.

1   12.   The computer system of claim 11, wherein while atomically

2   committing changes made during the transactional execution, the execution

3   mechanism is configured to:

4   treat store-marked cache lines as locked, thereby causing other processes

5   to wait to access the store-marked cache lines;

24

6   clear load marks from cache lines;

7   commit store buffer entries generated during transactional execution to

8 memory, wherein committing each store buffer entry involves unmarking, and

9 thereby unlocking, a corresponding store-marked cache line; and to

10   commit register file changes made during transactional execution.

1   13. The computer system of claim 10, wherein if an interfering data

2 access from another process is encountered during the transactional execution and

3 prior to encountering the commit instruction, the execution mechanism is

4 configured to:

5   discard changes made during the transactional execution; and to

6   attempt to re-execute the block of instructions.

1   14. The computer system of claim 10, wherein if a variation of the

2 commit instruction is encountered, the execution mechanism is configured to:

3   atomically commit changes made during the transactional execution; and

4 to

5   commence transactional execution of the block of instructions following

6 the commit instruction.

1   15. The computer system of claim 10, wherein the computer system is

2 configured to allow potentially interfering data accesses from other processes to

3 proceed during the transactional execution of the block of instructions.

1   16. The computer system of claim 10, wherein the block of

2 instructions to be executed transactionally comprises a critical section.

25

Attorney Docket No. SUN-P9324-MEG   Inventors: Tremblay et al.

ARP H:\SUN MICROSYSTEMS\SUN-P9324-MEG\SUN-P9324-MEG APPLICATION.DOC

1   17.   The computer system of claim 10, wherein the commit instruction
2   is a native machine code instruction of the processor.

1   18.   The computer system of claim 10, wherein the commit instruction
2   is defined in a platform-independent programming language.

1   19.   A computing means that supports a commit instruction to facilitate
2   transactional execution, wherein the commit instruction marks the end of a block
3   of instructions to be executed transactionally, comprising:
4   a processing means; and
5   an execution means within the processing means;
6   wherein upon encountering the commit instruction, the execution means is
7   configured to successfully complete transactional execution of the block of
8   instructions preceding the commit instruction;
9   wherein changes made during the transactional execution are not
10   committed to the architectural state of the processor until the transactional
11   execution successfully completes.

1   20.   The computing means of claim 19, wherein while successfully
2   completing transactional execution, the execution means is configured to:
3   atomically commit changes made during the transactional execution; and
4   to
5   resume normal non-transactional execution.

26